PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-315954
(43)Date of publication of application : 14.11.2000

(51)Int.Cl. H03M 7/30

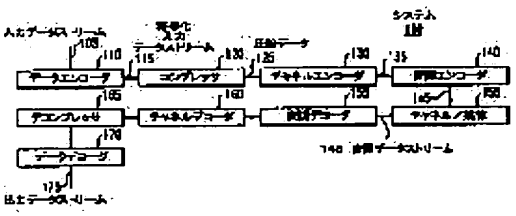
(21)Application number : 2000-078069
(22)Date of filing : 21.03.2000

(71)Applicant : LUCENT TECHNOL INC
(72)Inventor : BENTLEY JON LOUIS
MCILROY MALCOLM DOUGLAS

(30)Priority
Priority number : 99 273840 Priority date : 22.03.1999 Priority country : US

(54) METHOD FOR COMPRESSING INPUTTED DATA STREAM AND DEVICE THEREFOR

(57)Abstract:
PROBLEM TO BE SOLVED: To improve data compression to a larger input data stream by crossing the input data stream and comparing an intermediate fingerprint calculated from a specified character set of the input data stream with a previously calculated and stored fingerprint.
SOLUTION: The input data stream 105 is supplied to an input data encoder 110, the input data stream is preprocessed and encoded. The encoded input data stream 115 is compressed to compressed data 125 by a compressor 120. A data stream from a channel decoder 160 is decompressed by a decompressor 165, decoded by a data decoder 170 and an output data stream 175 is created. And the input data stream is divided into groups of blocks, after that, the fingerprints are calculated for respective blocks and stored. Data compression structure to recognize correlation between different strings is further used.



LEGAL STATUS

[Date of request for examination] 23.04.2002
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－315954
（P2000－315954A）
(43)公開日　平成12年11月14日(2000.11.14)

(51)Int.Cl.'	識別記号	FI		テーマコード'(参考)
H03M 7/30		H03M 7/30	Z	

審査請求　未請求　請求項の数38　ＯＬ　（全 17 頁）

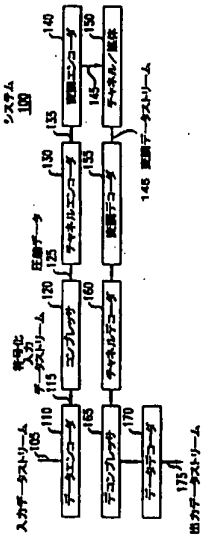
(21)出願番号　特願2000－78069(P2000－78069)
(22)出願日　平成12年３月21日(2000.3.21)
(31)優先権主張番号　０９／２７３８４０
(32)優先日　平成11年３月22日(1999.3.22)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　596077259
ルーセント　テクノロジーズ　インコーポレイテッド
Ｌｕｃｅｎｔ　Ｔｅｃｈｎｏｌｏｇｉｅｓ　Ｉｎｃ．
アメリカ合衆国　07974　ニュージャージー、マレーヒル、マウンテン　アベニュー　600－700
(74)代理人　100081053
弁理士　三俣　弘文

最終頁に続く

(54)【発明の名称】　入力データストリームの圧縮方法とその装置

(57)【要約】
【課題】　記憶装置の条件や伝送時間を減らすために更に大きく圧縮する方式を提供する。
【解決手段】　圧縮プロセス（例えば、いずれかのLempel-Ziv圧縮方式）を適用する前に入力データの初期評価として入力データストリームのより長い履歴およびより長いコモンストリングを用いる。すなわち、通常の圧縮プロセスが所望の圧縮を行うために比較的短い（最も最近の数キロバイト）入力データの履歴を用いるが、より長いコモンストリングシーケンスを用いることとより長い履歴を用いることとが組み合わさって、圧縮効率を増加できる。これは特に、繰り返す長いストリングを多く有する長い入力ストリームを圧縮する際に有効である。本発明に従うと、摘出した長コモンストリングに対してストリングマッチングを適用することになり入力データをプリプロセス（前処理）する。



【特許請求の範囲】
【請求項１】　入力データストリームを圧縮する方法であって、
（Ａ）入力データストリームを複数のデータブロックへと分割するステップと、
（Ｂ）複数のフィンガプリントを計算するステップと、ここで、各フィンガプリントは、前記複数のデータブロックの異なる１つのデータブロックに対応し、
（Ｃ）前記複数のフィンガプリントの各フィンガプリントは、前記複数のデータブロックの異なる１つのデータブロックに対応し、入力データストリームを前記複数のフィンガプリントと比較するステップと、
（Ｄ）特定のフィンガプリントと入力データストリームの間にマッチが出現すれば、そのマッチを入力データストリーム内のそのマッチを符号化するステップと、
（Ｅ）符号化したデータストリームを圧縮データストリームへと圧縮するステップとを有することを特徴とする方法。
【請求項２】　前記入力データストリームは一連の文字からなることを特徴とする請求項１記載の方法。
【請求項３】　前記ステップ（Ｃ）は、一連の文字の個々の文字の関数として入力データストリームを順に横断し、その個々の文字のそれぞれの関数として中間フィンガプリントを計算することを特徴とする請求項２記載の方法。
【請求項４】　（Ｆ）前記複数のフィンガプリントと共に前記複数のデータブロックをデータ構造に記憶するステップを更に有することを特徴とする請求項２記載の方法。
【請求項５】　前記ステップ（Ｃ）は、特定のフィンガプリントにマッチする一連の文字から最も長いマッチしたシーケンスを識別することを特徴とする請求項３記載の方法。
【請求項６】　前記ステップ（Ｃ）は、中間フィンガプリントを特定のフィンガプリントと比較することを特徴とする請求項３記載の方法。
【請求項７】　前記ステップ（Ｄ）は、最も長いマッチしたシーケンスの入力データストリームにおける開始位置および最も長いマッチしたシーケンスの長さを符号化することを特徴とする請求項５記載の方法。
【請求項８】　複数の文字を含む入力データストリームを圧縮コードストリームへと処理する方法であって、
（Ａ）入力データストリームを複数のブロックに分割するステップと、ここで、各ブロックは、複数の文字のうち特定の数の文字を含み、
（Ｂ）複数のフィンガプリントを計算するステップと、ここで、各フィンガプリントは、前記複数のデータブロックの異なる１つのデータブロックに対応し、
（Ｃ）各フィンガプリントは複数のブロックの異なる１つのブロックに対応し、特定のフィンガープリントと特定のブロックのいずれかの部分との間にマッチが出現するかを判断するために、複数のブロックを複数のフィンガプリントと比較するステップと、
（Ｄ）出現したマッチそれぞれに対して入力データストリームにて識別子を符号化するステップと、ここで、この識別子は、特定のブロックのマッチ部分の入力データストリームにおける開始位置およびマッチ部分の長さを含み、
（Ｅ）符号化された入力データストリームを圧縮コードストリームへと圧縮するステップとを有することを特徴とする方法。
【請求項９】　符号化された入力データストリームの圧縮は、Lempel-Zivコーディング技術に従って行われることを特徴とする請求項１、８記載の方法。
【請求項１０】　前記ステップ（Ｃ）は、
（ａ）ウインドウサイズを選択するステップと、
（ｂ）ウインドウサイズの関数として複数のブロックを横断するステップとを有することを特徴とする請求項８記載の方法。
【請求項１１】　ウインドウサイズは、特定の数の文字であることを特徴とする請求項１０記載の方法。
【請求項１２】　前記ステップ（Ｃ）は、
（ｃ）ブロックに含まれる特定の数の文字それぞれにおけるブロックに対する中間フィンガプリントを計算するステップと、
（ｄ）中間フィンガプリントを特定のフィンガプリントと比較するステップとを有することを特徴とする請求項１１記載の方法。
【請求項１３】　前記複数のブロックの各ブロックはブロックサイズが等しいことを特徴とする請求項１１記載の方法。
【請求項１４】　前記ブロックサイズは１０～１０００文字の範囲であることを特徴とする請求項１３記載の方法。
【請求項１５】　前記横断（ｂ）は、ブロックに含まれる特定の数の文字を順に横断することを特徴とする請求項１２記載の方法。
【請求項１６】　（Ｆ）前記複数のフィンガプリントと共に前記複数のデータブロックをデータ構造に記憶するステップを有し、このデータ構造は、各フィンガプリントをその異なる１つの対応するブロックと共に記憶することを特徴とする請求項８記載の方法。
【請求項１７】　前記開始位置は、入力データストリームにおける文字の位置であり、マッチした部分の長さは文字の数であることを特徴とする請求項１６記載の方法。
【請求項１８】　（Ａ）入力データストリームを受信し複数のブロックへと分割する受信器と、ここで、
各ブロックは、入力データストリームからの複数の文字の特定の数の文字を含み、

（B）複数のフィンガプリントを計算するエンコーダと、ここで、
各フィンガプリントは、前記複数のブロックの異なる１つのブロックに対応し、このエンコーダは、複数のブロックを複数のフィンガプリントと比較し、特定のフィンガプリントと特定のブロックのいずれかの部分の間でマッチが出現するかを判断し、出現した各マッチに対してそのマッチの入力データストリームにおける識別子を符号化し、
（C）符号化された入力データストリームを圧縮コードストリームへと圧縮するコンプレッサとを有することを特徴とするデータ圧縮を行う装置。
【請求項１９】　（D）複数のブロックを複数のフィンガプリントと共にデータ構造に記憶するメモリーを更に有し、
このデータ構造は、各フィンガプリントをその異なる１つの対応するブロックと共に記憶することを特徴とする請求項１８記載のデータ圧縮を行う装置。
【請求項２０】　前記複数のブロックの比較は、（a）ブロックに含まれる特定の数の文字の各文字におけるブロックに対して中間フィンガプリントを計算し、
（b）中間フィンガプリントを特定のフィンガプリントと比較することを特徴とする請求項１８記載のデータ圧縮を行う装置。
【請求項２１】　前記識別子は、特定のブロックのマッチした部分の入力データストリームにおける開始位置およびマッチした部分の長さを含むことを特徴とする請求項１９記載のデータ圧縮を行う装置。
【請求項２２】　前記複数のブロックの各ブロックは、ブロックサイズが等しいことを特徴とする請求項２１記載のデータ圧縮を行う装置。
【請求項２３】　前記開始位置は、入力データストリームにおける文字の位置であり、前記マッチした部分の長さは、文字の数であることを特徴とする請求項２２記載のデータ圧縮を行う装置。
【請求項２４】　前記コンプレッサ（C）は、圧縮コードストリームへと符号化された入力データストリームを圧縮するのにLempel-Zivコーディングを用いることを特徴とする請求項１８記載のデータ圧縮を行う装置。
【請求項２５】　デジタルデータストリームを圧縮データ形式で記憶するデータ記憶システムであって、
（A）デジタルデートストリームを複数のデータブロックへと分割する手段と、ここで、各ブロックは複数の文字を有し、
（B）複数のフィンガプリントを計算する手段と、ここで、各フィンガプリントは、前記複数のデータブロックの異なる１つのデータブロックに対応し、
（C）複数のブロックを横断する手段と、
（D）ブロックの複数の文字の各文字における各ブロックに対する中間フィンガプリントを計算する手段と、
（E）中間フィンガプリントを複数のフィンガプリントの特定のフィンガプリントと比較する手段と、
（F）その特定のフィンガプリントと中間フィンガプリントの間でマッチが出現すれば、そのマッチをデジタルデータストレームにおいて符号化する手段と
（G）符号化されたデータストリームを圧縮データ形式に圧縮する手段と、を有することを特徴とするデータ記録システム。
【請求項２６】　（H）複数のデータブロックを複数のフィンガプリントと共にデータ構造に記憶する手段を更に有することを特徴とする請求項２５記載のデータ記録システム。
【請求項２７】　前記横断する手段（C）は、各ブロックの複数の文字の関数としてブロックを順に横断することを特徴とする請求項２６記載のデータ記録システム。
【請求項２８】　前記圧縮する手段（G）は、符号化されたデジタルデータストリームを圧縮データ形式に圧縮するためにLempel-Zivコーディングを用いることを特徴とする請求項２７記載のデータ記録システム。
【請求項２９】　前記符号化されたマッチは、特定の部分のマッチした部分のデジタルデータストリームにおける開始位置およびマッチした部分の長さを含むことを特徴とする請求項２５記載のデータ記録システム。
【請求項３０】　圧縮デジタル信号を処理する装置であって、
圧縮デジタル信号は、
（a）入力デジタルデータストリームを複数のデータブロックへと分割し、
（b）それぞれが複数のデータブロックの異なる１つのデータブロックに対応する複数のフィンガプリントを計算し、
（c）入力デジタルデータストリームを複数のフィンガプリントと比較し、
（d）特定のフィンガプリントと入力デジタルデータストリームの間でマッチが出現すれば、入力デジタルデータストリームにおいてそのマッチを符号化し、
（e）符号化された入力デジタルデータストリームを圧縮デジタル信号に圧縮し、
（f）圧縮デジタル信号を通信チャネルへ、供給することにより作られ、
（A）通信チャネルから圧縮デジタル信号を受信する受信器と、
（B）受信した圧縮デジタル信号を脱圧縮し、その脱圧縮したデジタル信号から入力デジタルデータストリームを回復するデコンプレッサとを有することを特徴とする圧縮デジタル信号を処理する装置。
【請求項３１】　前記複数のブロックの各ブロックは、複数の文字を含み、前記入力デジタルデータストリームの比較（c）は、
（g）ブロックの複数の文字の各文字におけるブロックに対する中間フィンガプリントを計算し、

（h）中間フィンガプリントを特定のフィンガプリントと比較することを特徴とする請求項３０記載の装置。
【請求項３２】　前記複数のブロックの各ブロックは、ブロックサイズが等しいことを特徴とする請求項３１記載の装置。
【請求項３３】　前記複数のフィンガプリントの各フィンガプリントは対応するデータブロックの特定の文字群を表すことを特徴とする請求項３１記載の装置。
【請求項３４】　以下の命令を含む複数の命令が記憶された機械が読み取り可能な媒体。すなわち、
（A）機械により実行されると、入力データストリームを圧縮コードストリームへと処理させ、
（B）入力データストリームを複数のブロックへと分割させ、ここで、各ブロックは複数の文字の特定の数の文字を含み、
（C）複数のフィンガプリントを計算させ、ここで、各フィンガプリントは、前記複数のブロックの異なる１つのブロックに対応し、
（D）複数のブロックを複数のフィンガプリントと比較させ、
（E）特定のフィンガプリントと特定のブロックのいずれかの部分の間にマッチが出現するかどうかを判断させ、
（F）出現した各マッチに対して、入力データストリームにおけるそのマッチの識別子を符号化させ、ここで、この識別子は、特定のブロックのマッチした部分の入力データストリームにおける開始位置およびマッチした部分の長さを含み、
（G）符号化された入力データストリームを圧縮コードストリームへと圧縮させる命令。
【請求項３５】　前記複数のフィンガプリントの各フィンガプリントはその対応するブロックの特定の文字群を表すことを特徴とする請求項３４記載の機械が読み取り可能な媒体。
【請求項３６】　前記比較（D）は、ブロックにおける文字の関数として行われ、ブロックに対する中間フィンガプリントは、前記複数の文字の各文字にて計算され、前記特定のフィンガプリントと比較されることを特徴とする請求項３４記載の機械が読み取り可能な媒体。
【請求項３７】　前記圧縮は、Lempel-Zivコーディング技術に従って行われることを特徴とする請求項３６記載の機械が読み取り可能な媒体。
【請求項３８】　（A）それぞれが複数の文字からなる複数のブロックへと入力データストリームを分割し、
（B）それぞれが前記複数のブロックの異なる１つのブロックに対応する複数のフィンガプリントを計算し、
（C）ブロックの複数の文字の各文字の関数として入力ストリームを横断し、
（D）ブロックの各文字におけるブロックに対する中間フィンガプリントを計算し、
（E）中間フィンガプリントを複数のフィンガプリントと比較し、
（F）複数のフィンガプリントの特定のフィンガプリントと中間フィンガプリントの間でマッチが出現するかどうか判断し、
（G）出現した各マッチに対して、ブロックのマッチした部分の入力データストリームの開始位置およびマッチした部分の長さを含む、入力データストリーム内の識別子を符号化し、
（H）符号化された入力データストリームを圧縮コードストリームへと圧縮し、
（I）圧縮コードストリームを表す記録信号を記憶媒体に供給し、
（J）記録信号を記憶媒体上に記録する動作からなるプロセスを用いて作成された記憶媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、データ記憶および通信システムに関し、特に、それらシステムの容量および利用を改善することに関する。
【０００２】
【従来の技術】従来のデータ圧縮技術およびシステムはデジタルデータストリームを圧縮コードストリームへと符号化し、圧縮コードストリームを対応する元のデータストリームへとデコードして戻す。ここでコードストリームを「圧縮」としているが、それは、コードストリームが通常、元のデータストリームに含まれる符号の数よりも少ない数のコードであるからである。このような小さいコードは元のデータよりも少ない量のメモリーに記憶することができる。
【０００３】また、圧縮コードストリームは、圧縮されていない元のデータよりも短い時間で通信システム（例えば、有線、無線、光ファイバ通信システム）にて送信することができる。今日の通信ネットワークにおいて、コンテンツ交換の量が相当に増大しており、データ送信および記憶容量の必要性が今までかつてないほど増えている。このように、データ圧縮は現代の送信プロトコルおよび通信ネットワークにおいて重要な役割を担っている。
【０００４】データ圧縮に有用な２つのクラスの圧縮技術は、周知のように、いわゆる特殊用途圧縮と汎用圧縮である。特殊用途圧縮技術は特殊な種類のデータを圧縮するために設計され、実装するのに比較的低コストであることが多い。例えば、周知な特殊用途圧縮技術として、ランレングス符号化、ゼロサプレッション符号化、ヌル圧縮符号化、パターン置き換えの技術がある。
【０００５】これら技術により、一般に、比較的小さな圧縮比となる。なぜなら、一般的な特性および冗長性を有するデータを通常圧縮するからである。圧縮比とは、

元のデータの長さに対しての圧縮コードの長さの測定値である。しかし、特殊用途圧縮技術はより一般的性質のデータ（共通な性質を多く有さないようなデータ等）を圧縮するのにあまり有効でないことが多い。

【０００６】対照的に、汎用圧縮技術は一種類のデータを特別に圧縮するのに設計されてはおらず、実際の圧縮プロセスの際に異なる種類のデータに適合することが多い。最も有名で有用な汎用圧縮技術は、Lempel-Zivコーディングとして知られる。J. ZivとA. Lempelによって開発された一連のアルゴリズムから派生している。これについて、文献、Ziv et al., 'A Universal Algorithms for Sequential Data Compression', IEEE Transactions on Information Theory, IT-23(3):337-343, May 1977（ＬＺ１アルゴリズムを記載している）、Ziv et al., 'Compression of Individual Sequences Via Variable-Rate Coding', IEEE Transactions on Information Technology, IT-24(5):530-536, September 1978（ＬＺ２アルゴリズムを記載している）に記載されている。これらＬＺ１、ＬＺ２データ圧縮方式は周知であり詳細には説明しない。

【０００７】ＬＺ１（ＬＺ７７とも呼ばれる）データ圧縮プロセスは繰り返された文字シーケンス（本明細書において、文字(character)とはビット群をいい言語としての文字を意味するのではない）は、そのシーケンス（マッチングシーケンス）の前の出現を参照することによって置き換えることができるという原理に基づいている。この参照（例えば、ポインタ）は通常、前の出現の位置（例えば、繰り返しシーケンスの開始からのオフセットバイト）、および繰り返す文字の数（マッチドレングス）を示す情報を含む。この参照は通常、伝統的なＬＺ１コーディングに従うと「<Offset, length>」として表されている。

【０００８】対照的に、ＬＺ２（ＬＺ７８としても知られる）圧縮は、圧縮の際に作られるルックアップテーブルないし辞書を適応的に成長させることに基づいて、入力データ文字のストリームを符号化値へとパースする（parse）。すなわち、ＬＺ２は、ＬＺ１コーディングのようにバイト境界や長さにおいて一致性を見つけたりはせず、辞書のワードがソースストリング（文字列）と一致すれば、新しいワードが辞書に加えられる。この辞書はその一致したワードとその後のソースストリングバイトからなる。ＬＺ２コーディングに従うと、一致は辞書内のワードへのポインタないしインデックスとして符号化される。

【０００９】上述のように、ＬＺ１およびＬＺ２アルゴリズムが用いる基本的な原理から派生した圧縮方式は多い。例えば、Terry A. WelchはＬＺ２コーディングを改善して周知のLempel-Ziv-Welch（ＬＺＷ）圧縮プロセスを作った。これについて、文献、T. A. Welch, 'A Technique for High performance Data Compression', IEEE Computer, pp. 8-19, June 1984および米国特許第４５５８３０２明細書（１９８５年１２月１０日発行）を参照するとよい。

【００１０】これらＬＺ２およびＬＺＷの両方の圧縮技術は入力文字のストリングを固定長コードへとマッピングするいわゆるストリングテーブルを生成し利用する。具体的には、これら圧縮技術は文字ストリームを順に検索しテーブル（辞書）に前に記憶した最長のストリングとマッチ（合致）する遭遇したシンボルのシーケンスに基づいてコードを生成することにより、データ文字のストリームを圧縮されたコードストリームへと圧縮する。各マッチがなされコードシンボルが生成されるに従い、このプロセスは辞書に新しいストリングエントリーをも記憶する。これは、データストリームにおけるマッチしたシーケンスに加えて、データストリームにおいて遭遇する次の文字シンボルからなる。

【００１１】上述のように、Lempel-Zivコーディングの要点は、元のデータストリーム（例えば、送信されるドキュメント）において繰り返されるストリングやサブストリングを見つけることである。圧縮されるドキュメントにおける繰り返されたフレーズは元のデータストリームにおいて前に出現した場所へのポインタによって置き換えられる。従って、この方法で圧縮されるデータ（例えば、テキスト）をデコードすることには、ポインタが指すすでにデコードしたテキストでポインタを置き換えることを必要とする。

【００１２】周知のように、Lempel-Zivコーディングを用いる際に設計上主に考慮することとして、ポインタがどれくらい戻るか制限を設定し、その制限をどのようにするか決めることがある。別の設計上に考慮することとして所望の制限内のどのサブストリングがポインタのターゲットとするかということがある。すなわち、前のテキストへのポインタのリーチの制限がなくなったり、いわゆるグローイングウィンドウ（growing window）、あるいは前の「N」文字の固定サイズのウィンドウに制限される。ここで、Nは通常、数千文字（例えば、３キロバイト）の範囲である。

【００１３】このコーディングに従うと、ストリングの繰り返しは、両方のストリングがウィンドウに出現した場合にのみ発見され圧縮される。このようなLempel-Zivコーディングの設計時には、速さ、メモリー条件、圧縮比の間で妥協して決められる。ウィンドウをスライドすることには少なくとも１つの欠点がある。ウィンドウをスライドさせる方式は、入力テキストにおいて遠くに出現するストリング（例えば、１００００文字分）を見つけることはできない。

【００１４】

【発明が解決しようとする課題】上述のようなＬＺ１、ＬＺ２、ＬＺＷ圧縮方式のような従来の圧縮方式は有効なデータ圧縮を提供しているが、記憶装置の条件や伝送

時間を減らすために更に大きく圧縮する方式が望まれている。

【００１５】

【課題を解決するための手段】本発明は、比較的低い圧縮比を実現する方法および装置を提供する。これは、圧縮プロセス（例えば、いずれかのLempel-Ziv圧縮方式）を適用する前に入力データの初期評価として入力データストリームのより長い履歴およびより長いコモンストリングを用いることを我々が認識したことに基づいている。すなわち、通常の圧縮プロセスが所望の圧縮を行うために比較的短い（最も最近の数キロバイト）入力データの履歴を用いるが、より長いコモンストリングシーケンスを用いることとより長い履歴を用いることとが組み合わさって、圧縮効率を増加できることを我々は認識した。これは特に、繰り返す長いストリングを多く有する長い入力ストリームを圧縮する際に有効である。

【００１６】本発明に従うと、摘出した長コモンストリングに対してストリングマッチングを適用することになり入力データをプリプロセス（前処理）する。好ましい実施例に従うと、入力データは、個々のブロックが均等なサイズ（線文字長）を有するように一連のブロックへと分割される。また、好ましい実施例に従うと、各ブロックに対していわゆるフィンガプリント（指紋）が計算され記憶される。フィンガプリントは、大きいテキストストリングの比較的小さいシグネチャーである。例えば、千文字のストリングが３２ビット長のフィンガプリントへとマッピングされる。従って、同一のストリングは常に同じフィンガプリントを有する。また、等しくないストリングはほとんど常に等しくないフィンガプリントを有する（特定の確率ファクタ内で）。

【００１７】本発明に従うと、入力データストリームが横断され（traverse）、入力ストリームの特定の文字セットから計算した（文字毎ベースで）中間フィンガプリントと、前に計算し記憶したフィンガプリントとの間で比較がなされる。好ましい実施例に従うと、入力ストリームは、均等なブロックサイズを有するスライドウィンドウの関数として横断され、中間フィンガプリントは現在の文字ウィンドウから計算され、前に記憶したフィンガプリントと比較される。

【００１８】好ましい実施例に従い、フィンガプリントの間にマッチを検出すると、入力ストリームは、検出したマッチの関数として決められた識別子とともに符号化される。符号化された識別子は、元の入力ストリームにおけるマッチングストリングの開始位置およびストリング長を含む。その後に、好ましい実施例に従うと、前処理され符号化された入力ストリームに対し更にLempel-Ziv圧縮を用いて圧縮がなされる。

【００１９】本発明に従うと、全体の記憶容量の条件を余り大きくせずに、長いコモンストリングを識別し、入力データの大きな履歴を調べることができる。本発明に従って、様々な圧縮方法によって大きな圧縮比を実現することができる。すなわち、本発明の原理はいずれの特定の圧縮方式には依存せず、広い範囲の圧縮方式において本発明の様々な原理を用いる利点を発揮することができる。

【００２０】ストリングマッチングメカニズムとしてフィンガプリントを用いることは新しくはない。テキスト処理システムにおけるストリングマッチングに対してフィンガプリントは用いられている。具体的には、テキストファイルにおいて長いコモンストリングを検索する際に用いられている。例えば、文献、R. M. Karp and M. O. Rabin, 'Efficient Randomized Pattern-Matching Algorithms', IBM J. Res. Develop., Vol. 31, No. 2, pp. 249-260, March 1987は、ストリング検索の際にフィンガプリントを用いることを記載している。しかし、我々はフィンガプリントを優雅な圧縮ツールとして導入できることを認識した。これにより、全体の記憶容量条件を余り大きくせずに、繰り返す長いストリングを多数有する。大きい入力データストリームに対してデータ圧縮を改善することができる。

【００２１】

【発明の実施の形態】圧縮プロセス（例えば、Lempel-Ziv圧縮方式のいずれか）を適用する前に入力データの初期評価として入力データストリームのより長い履歴およびより長いコモンストリングを用いることの認識に基づいて、本発明は比較的圧縮比を実現する方法および装置を提供する。通常の圧縮プロセスが所望の圧縮をするために入力データの比較的短い履歴（最も最近のバイト群）を用いるが我々はより長いコモンストリングシーケンスと共により長い履歴を用いることにより、特に、繰り返す長いストリングを多数有するより長い入力ストリーム（例えば、大規模データベース）を圧縮する際に、圧縮効率を増加することができることを認識した。

【００２２】本発明は、これら方法を実践する方法および装置の形態で実現することができる。また本発明は、ＦＤ（ｆｌｏｐｐｙ（登録商標）　ｄｉｓｋｅｔｔｅ）、ＣＤ－ＲＯＭ、ハードディスクドライブ、機械が読み取り可能な記憶媒体のような実体的な媒体に実装されるプログラムコードの形態にて実現することもできる。この場合、プログラムコードが機械（例えば、コンピュータ）へとロードされ機械によって実行されると、その機械が本発明を実行する装置となる。また、本発明はプログラムコードの形態にて実装することができ、例えば、機械へとロードされおよび／または機械によって実行され、あるいは電線、ケーブリング、光ファイバ、電磁放射のような何らかの伝送媒体にて送信されるようなプログラムコードの形態にて実装される。プログラムコード汎用プロセッサに実装された場合に、プログラムコードセグメントがプロセッサと組合わさって、特定のロジック回路と類似するように動作するユニークなデバ

イスを与える。

【００２３】図１は、本発明に従ってデータを圧縮したり脱圧縮したりするためのシステム１００のブロック図である。システム１００は、ほんの少しだけ名前を挙げるのに伝送媒体（例えば、有線、無線、光ファイバ等）上に情報を送信するのに用いられる。また、システム１００は、例えば、コンピュータのディスクドライブのような磁気媒体、ＣＤ－ＲＯＭのような光学的に読み取り可能な媒体、インターネット上の媒体へと情報を記録し、またはそれらから情報を読み取るのにも有用である。

【００２４】従って、本発明に従って圧縮されたデータを磁気ディスクドライブのような磁気媒体、ＣＤ－ＲＯＭのような光記録可能な媒体上に記録することができる。図１において、入力データストリーム１０５（例えば、テスト）が入力データエンコーダ１１０に供給される。下で詳細に述べるように、本発明に従う入力データエンコーダ１１０は、抽出する長いコモンストリングに対してストリングマッチング技術を適用することにより本発明に従って入力データストリームをプリプロセスおよび符号化する。この符号化プロセスに関連する本発明の多くの原理は下で図２に示した例を特に参照してより広く説明する。

【００２５】図１に戻ると、本発明に従って作られた符号化入力データストリーム１１５はコンプレッサ１２０に渡される。好ましい実施例に従うコンプレッサ１２０は、符号化入力データストリーム１１５を圧縮データ１２５に圧縮するのにLempel-Ziv圧縮を、具体的には、ＬＺ７７コーディングを適用する。ここで、上述のように本発明に従って符号化入力データストリーム１１５を圧縮するのにいずれのLempel-Zivタイプの圧縮をも有効に用いて本発明の原理の利点を実現することができることに留意されたい。

【００２６】次に圧縮データ１２５はチャネルエンコーダ１３０により符号化されチャネル符号化された情報１３５を作る。チャネル符号化により圧縮された情報に対し情報を加え、エラー検出やデータ読み取りプロセスにおける訂正を可能にする。伝統的なチャネル符号化技術としては、各シンボルが１もしくは複数のデータビットで表されるシンボルのシーケンスを符号化する周知のReed-Solomon符号化がある。次にこれらシンボルは変調エンコーダ１４０により変調符号化され、変調されたデータストリーム１４５を作る。これは、伝送チャネルを通って送信されるかあるいは媒体１５０上に記録されるチャネルシーケンスを定める。

【００２７】ノイズや干渉は多くの場合、データストリームの伝送や記録時にチャネル／媒体１５０にて投入される。従って、変調デコーダ１５５、チャネルデコーダ１６０はノイズとともに変調データストリーム１４５を受信し、周知な方法で、チャネルエンコーダ１３０、変調エンコーダ１４０の符号化プロセスをそれぞれ逆にたどる。チャネルデコーダ１６０からのデータストリームはコンプレッサ１２０が生成する圧縮データ１２５に対応する。図７、１１に関連して詳細に説明するように本発明に従って、このデータストリームはデコンプレッサ１６５により脱圧縮され、データデコーダ１７０によりデコードされ、出力データストリーム１７５を作る。

【００２８】本発明の多くの原理は、圧縮による相当な削減や伝送効率を実現することに関連している。図２は、本発明に従ってデータを圧縮する動作を示す流れ図であり、図１のシステムにて有用である。入力データストリーム（例えば、テキストファイル）が受信される（２１０）。特定の圧縮ブロックサイズ「ｂ」が選択される（２２０）。このｂは特定の文字の数である。好ましい実施例に従って、ｂは２０～１０００文字の範囲で選択される。入力データストリームはサイズｂのブロック群に分割される（２３０）。その後に、本発明に従って、ブロック群の各ブロックに対しフィンガプリントが計算され格納される（２４０）。

【００２９】好ましい実施例に従って、文献、Karp et al., supra. に記載された技術に従ってフィンガプリントが計算される。カープらはストリング検索を支援するのにフィンガプリントを元々用いた。すなわち、長さｎのストリングが長さｍのサーチパターンを含むかどうかである。カープらはサーチパターンのｍの文字をポリノミアルモジュロ（多項式法処理）した大きな素数として解釈した。従って、得られるフィンガプリントは、例えば、３２ビットワードとして格納することができる。カープの技術は入力ストリングを走査し、長さｍのｎ－ｍ＋１のサブストリングのそれぞれに対し同じフィンガプリントを計算する。もしこれらフィンガプリントがマッチしなければ、そのサブストリングはパターンにマッチしないという結論を出す。もしそれらフィンガプリントがマッチすれば、そのサブストリングは実際にはパターンにマッチするかどうかの更なるチェックを行う。

【００３０】カープらの技術は以下のようなフィンガプリントの幾つかの有用な特性を証明した。すなわち、

（１）フィンガプリントを迅速に計算することができること。すなわち、フィンガプリントをＯ（ｍ）の時間で初期化することができ、Ｏ（１）の時間である位置をスライドすることにより更新することができること。

（２）フィンガプリントは偽のマッチを得ること。すなわち、等しくないストリングは等しくないフィンガプリントを持つことが非常に確かであること。（２つの等しくないストリングが同じ３２ビットフィンガプリントを有する確率は約2^{-32}である。）

（３）大きな素数をランダムで選ぶことができ、テキストストリング検索においてランダム化したアルゴリズムを得ることができること。

【００３１】我々は、繰り返しストリングを有する非常に大きな入力データストリームに対し増強したデータ圧縮を行うようなエレガントな圧縮ツールを導入することによってフィンガプリントを用いることを認識した。本発明に従うと、全体の記憶容量の必要条件の余り増加させないで長いコモンストリングを識別して入力データの大きな履歴を調べることができる。すなわち、本発明に従って、異なるストリングの間の相関を認識するデータ圧縮構成を用いる。具体的には、特定のテキストストリングの第２出現を繰り返しとして認識し、この第２出現を符号化した第１ストリングへの参照によって置き換える。従って、本発明に従うと、多くの圧縮方法に対し大きな圧縮比を実現することができる。

【００３２】特定の圧縮方式におけるストリング繰り返しの認識は知られている。例えば、文献、J. G. Cleary et al., "Unbounded length contexts for PPM", ComputerJournal 40.2/3, pp. 67-75, 1997, C. G. Nevill-Manning et al., "A space-economical suffix tree construction algorithm", Journal of ArtificialInteligence Research, pp. 67-82, September 1997にはストリング繰り返しを認識している特定の圧縮方式を記載している。しかし、本発明とは対照的にこれらの従来の方式は大きな量のメモリーを必要とする。すなわち、フィンガプリントを用いない方法において、ｎ文字のファイルを処理をするのに約ｎのワードを主メモリーに必要とする。下で議論するように、記憶条件の必要条件を激的に増加せずに本発明に従うと大きな圧縮比を実現することができる。

【００３３】図２において、好ましい実施例に従うと、各ブロックに対し計算したフィンガプリントを格納する（２４０）。入力ストリームの各ブロック境界にてフィンガプリントが記録される。また、プライマリデータ構造がｂバイトの重なり合わないブロックのそれぞれのフィンガプリントを記憶する。すなわち、好ましい実施例に従うと、各バイト１...ｂ、ｂ＋１...２ｂ、２ｂ＋１...３ｂ等々のようにフィンガプリントが記憶される。

【００３４】好ましい実施例に従うと、約ｎ／ｂのフィンガプリントが記憶される。このｎは上述のストリングの長さである。本発明に従うと、元の入力ストリームの少ない割合のみが記憶され、従って、記憶装置の必要条件を低く押さえることができる。また、好ましい実施例に従うと、入力データストリームにおけるそのシーケンスの位置を表す正数と共にお互いをハッシュテーブル（周知なデータ構造）にてフィンガプリントを記憶し表現する。

【００３５】図３は、図２の動作によって計算されたフィンガプリントを記憶するデータ構造３００を示す。データ構造３００は入力ストリームの各ブロックを記憶する。例えば、ブロック１～ｍはブロック３０５～３２５として示した。それに加え、各ブロックに対し、計算したフィンガプリントを記憶する。例えば、図３においてＦＰ１～ＦＰｍをフィンガプリント３３０～３５０として示した。更に、下で述べるように、入力データストリーム（データ構成３００）を横断するのにスライディングウィンドウ３５５を用い、上述のマッチを検出するのに現在の文字のウィンドウを記憶されたフィンガプリントと比較する。

【００３６】より詳細には、図２において、入力データストリームを横断し、入力文字と記憶されたフィンガプリントの間に比較を行い（２５０）、マッチを検出する（２６０）。すなわち、入力データストリームを横断するのにスライディングウィンドウ（例えば、スライディングウィンドウ３５５）を用い、現在の文字のウィンドウに対していわゆる「中間(interim)」フィンガプリントを計算する。これは図においては現在の文字のウィンドウで文字毎のベースで行われている。

【００３７】これら現在の文字のウィンドウにわたって計算される中間フィンガプリントはマッチを検出するために格納フィンガプリントと比較される。具体的には、入力テキストを走査するに従って、コモンフィンガプリントを見つけるためにハッシュテーブルが用いられ、コモンストリングの位置を判断する。もしマッチが出現すれば、シーケンス「<start, length>」を用いてそのマッチを符号化する（２７０）。ここで、startは初期位置であり、lengthはコモンシーケンスのサイズである。

【００３８】例えば、以下の入力データストリームを考えてみる。

The Constitution of the United States, PREAMBLE We, the people of theUnited States, in order to form a more perfect union...

上述のような本発明の原理を適用すると、以下の符号化データストリームの結果を得ることができる。

The Constitution of the United States, PREAMBLE We, the people <16,21>, in order to form a more perfect union...

この符号化データストリームから、繰り返しストリングは'the people of theUnited States'は上述のように本発明の'<start, length>'シーケンスフォーマットを用いる識別子で符号化されていることがわかる。

【００３９】好ましい実施例に従うと、マッチするフィンガプリントを有するブロックがいわゆる偽マッチではないことを確実にするため、可能な限りであるが長さがｂ－１文字よりも長くない範囲で、（入力データストリームにわたって）逆方向および順方向でマッチの拡張を行う。もし幾つかのブロックが現フィンガプリントとマッチすれば、そのようなブロックの中での最大のマッチが本発明に従って符号化される。

【００４０】フィンガプリントに対しての入力文字（中間フィンガプリント）の比較は入力ストリームの終わりに到達するまで継続する（２８０）。その後に、符号化

データストリーム（本発明に従って符号化された元の入力データストリーム）が周知の圧縮アルゴリズムのいずれか（例えば、Lempel-Ziv圧縮）を圧縮される（２９０）。

【００４１】以下の擬コードは、上述のような本発明の原理に従うフィンガプリントの比較および符号化を記したものである。変数fpは、フィンガプリントを表し、関数checkformatchはハッシュテーブルにおけるフィンガプリントをルックアップし、マッチを見つければそのマッチを符号化する。

コード１

```
initialize fp
for(i=b; i<n; i++)
if(i%b == 0)
store(fp, i)
a[i]を含みa[i-b]を含まないようにfpを更新
checkformatch(fp, i)
```

【００４２】上の擬コードは、本発明を実装するプロセッサにおける実行をするために、多くのプログラム（例えば、Ｃ言語プログラム）にて開発するのに用いることができる。例えば、図９～１０は、本発明に従ってデータ圧縮するＣ言語ソースプログラム９００を示す。ソースプログラム９００は、このソースコードプログラム９００全体で用いる特定のデータ型、変数、データ構造を規定するプログラム命令を含むプログラムソースコード部分９１０を含む。

【００４３】プログラムソースコード部分９２０は、上述のように本発明に従ってストリングマッチング動作を実装するプログラム命令を含む。プログラムソースコード部分９３０は、上述のように本発明に従って計算されたフィンガプリントを格納するのに用いるハッシュテーブルデータ構造を定め構築するのを支援するプログラム命令である。プログラムソースコード部分９４０、９５０（図１０）は、上述のように本発明に従って符号化データファイルを作るために圧縮を完成させるプログラム命令を含む。

【００４４】Ｃ言語ソースプログラム９００は、生来的に例示的であり本発明を理解するのを支援するために表現してある。本発明を具現化する他のプログラムを本発明の範囲から外れずに開発することは当業者であればできるであろう。

【００４５】上述の本発明の多くの原理を更に説明するため、図４は、一連の短い非圧縮入力データストリーム４００を、本発明に従って符号化された対応する一連の符号化出力データストリーム４１０と共に示してある。図４を調べることにより、入力データストリーム４２０～４５０はそれぞれ、符号化出力４６０～４９０それぞれに示すようにマッチングストリングの符号化された表現によって処理されている。

【００４６】また、図５、６には本発明を更に説明するために、大きな入力データファイル、具体的には、アメリカ合衆国の憲法に対する本発明の原理の適用を示している。図５は、憲法の選択部分からなる入力データファイル５００を示している。本発明に従う憲法のテキストの圧縮は、同じ長いストリングが頻繁に現れることを考えると特に本発明の利点を発揮できる。例えば、入力データファイル５００のテキスト部分５１０、５２０、５３０は幾つかの長い繰り返すストリングを含む。

【００４７】従って、上述の本発明の多くの原理を適用すると、図６に示した符号化データファイル６００を得る。６００は符号化部分６１０、６２０、６３０を有し、これらそれぞれは入力データファイル５００のテキスト部分５１０、５２０、５３０に対応する。符号化部分６１０、６２０、６３０はそれぞれ、エンコーディング（例えば、エンコーディング６３５－６９０）を含む。これらは、本発明に従って導かれ、ブロックサイズ＝２０を用い、更に圧縮比を増しデータ伝送レートをより効率的にするために更に圧縮させることができる。

【００４８】例えば、エンコーディング６３５は、文字位置３９１に始まる４７文字のマッチングシーケンス（すなわち、マッチングストリング）が検出され符号化されたことを示す。同様に、例えば、エンコーディング６７５は、文字位置２４３９に始まる１０３文字のマッチングシーケンスが検出され符号化されたことを示す。

【００４９】圧縮プロセスを適用する前に入力データの初期評価（プリプロセス）においてフィンガプリントを用いることを認識することにより、入力データストリームの長い履歴および長いコモンストリングを用いることに基づいて比較的低い圧縮比を実現することができる。重要なことに、本発明は記憶装置の必要条件を余り増加させず、いずれの特定の圧縮技術にも依存しない。すなわち、本発明は、相当な圧縮比を実現する多様な周知の圧縮アルゴリズムを用い、記憶装置の必要条件や転送時間を減らすことができる。

【００５０】圧縮テスト結果を記す前に、本発明を用いて得られる符号化データファイルの脱圧縮を瞥見する。図７は、本発明の更なる原理に従ってデータを脱圧縮する動作７００の流れ図である。本発明による符号化データファイル（例えば、符号化データファイル６００）から、この態様では文字毎のベースで個々の文字「ｃ」を取り出す（７１０）。

【００５１】特定の文字ｃがシンボル「<」とマッチするかどうかの判断を行う（７２０）。マッチしなければ、脱圧縮プロセスに従って出力ファイルへと文字ｃが書き込まれ（７３０）、符号化データファイルから次の文字が取り出される。マッチすれば、符号化データファイルから次の文字が取り出される（７４０）。再び、特定の文字ｃがシンボル「<」とマッチするかどうかの判断をする（７５０）。マッチすれば、出力ファイルにシンボル「<」が書き込まれ（７６０）、次の文字が取り出される。マッチしなければ、上で詳細に示した「<start, length>」エンコーディングが符号化ファイルから読み取られる（７７０）。

【００５２】この態様に従うと、<start, length>情報を用いて、「start」位置から始まり「length」情報と等しい長さを有する符号化ファイルから出力ファイルへとストリングがコピーされる（７８０）。すなわち、本発明に従って前に符号化されたマッチングストリングがそれ全体で脱圧縮され出力ファイルへと書き込まれる。文字が残っていれば（７９０）、脱圧縮プロセスは継続する。残っていなければ、脱圧縮プロセスは完了し出力ファイルを完成させる。

【００５３】本発明の脱圧縮プロセスは、コンピュータ、プロセッサ、ＤＳＰ等で実行させることができる。図１１は、図７に示した本発明に従うデータを脱圧縮する動作を実装するＣ言語ソースコードプログラム１１００を示す。ソースコードプログラム部分１１００は符号化データファイルおよびそこに含まれる個々の文字を操作する配列を定める。部分１１２０は文字を処理するプログラムの主要部分を実行し、符号化ファイルを脱圧縮して出力ファイルを作る。

【００５４】Ｃ言語ソースコードプログラム１１００は生来的に例示的であり、本発明の理解を深めるために記した。本発明の原理を実装する他のプログラムを本発明の範囲を外れずに当業者は容易に作ることができるであろう。

【００５５】本発明の多くの原理および利点を説明するため、非常に大きなファイルに関連して本発明を適用し、そのファイルの圧縮の多くの結果を比較した。サンプルとしては、周知のＣＤ－ＲＯＭで"Project Gutenberg Compact Disc", Walnut Creek CDROM, Walnut Creek, CA に含まれる全てのテキストファイルを連結させて用いた。このＣＤ－ＲＯＭには１９９４の文書が含まれている。この試験のため、我々は周知のＵＮＩＸ（登録商標）の「cat」コマンドを用いて全てのテキストファイルを連結した。具体的なＵＮＩＸコマンド文字列は、「cat */*.txt > gut94all.txt」であった。このように連結することにより、６６１２２MBytesの入力ファイルを得て、これに本発明の原理を適用した。

【００５６】図８は、連結した当該テキストファイルを圧縮した結果の比較８００を示す。ファイルサイズはMByteで示してあり、圧縮による元ファイルと割合をパーセントで示した。ブロックサイズｂ（８１０）を変化させた効果を比較した。「comb」の見出しを有する列８２０は、本発明を適用した効果を示してあり、入力テキストファイルのサイズの変化を示してある。「com %」の見出しを有する列８３０は、ブロックサイズを変化させて調べた圧縮パーセンテージを示す。

【００５７】「com b|gzip」の見出しを有する列８４０および「com|gzip %」の見出しを有する列８５０は、本発明の原理と組み合わせて周知の「gzip」圧縮アルゴリズム（周知のＧＮＵによるＬＺ７７の実装であるgzip）を適用した結果を示す。「com|gzip %」（８５０）は、列８２０のサイズに対する列８４０の割合を示す。例えば、線のブロックサイズを用い本発明と組み合わせてgzipを適用すると、元のファイルパーセントを１９．７％の割合で減らすことができた。図よりブロックサイズbが減ると、最適でない点（効率的な圧縮にはブロックサイズが小さすぎる）に到達するまで圧縮の度合いは増えている。また、「total %」の見出しを有する列８６０は、元のファイルと比較した列８４０の割合を示してあり、bの最適な選択は、comがgzipのファイルサイズの減少量を超えて２２．５％のファイルサイズの減少量となっているところの３１MByteである。

【００５８】本明細書における説明は単に本発明の原理の例を示したのみである。当業者は、本発明の範囲を外れずに多くの構成を変えずに本明細書が開示することに基づいて多くの構成を考えることができるであろう。特許請求範囲の記載において特定の機能を実行する手段として表現したいずれの要素も、その機能を実行するいずれの機能をも表すように意図してある。例えば、（ａ）その機能を実行する回路要素の組み合わせ、あるいは（ｂ）その機能を実行するためソフトウェアを実行する適切な回路と組合わさるいずれの形態におけるソフトウェア（従って、ファームウェア、オブジェクトコード、マイクロコード等をも含む。）を含む。

【図面の簡単な説明】

【図１】本発明に従ってデータを圧縮し展開するシステム。

【図２】本発明に従ってデータを圧縮する動作の流れ図。これは図１のシステムにて有用である。

【図３】図２の動作に従って計算された入力データストリームやフィンガプリントを記憶するデータ構造の例。

【図４】本発明に従って符号化された一連の符号化データストリームとともに一連の非圧縮データストリームを示す。

【図５】入力データファイルの選択部分。

【図６】本発明の原理に従って図５の入力データファイルから符号化された符号化データファイルの選択部分。

【図７】本発明に従い展開する動作の流れ図。

【図８】本発明に従ってテキストファイルを圧縮した圧縮結果。

【図９】図２に示すように本発明に従ってデータを圧縮するＣ言語ソースコードプログラム。

【図１０】図９と同様。

【図１１】図７に示すように本発明に従ってデータを展開するＣ言語プログラム。

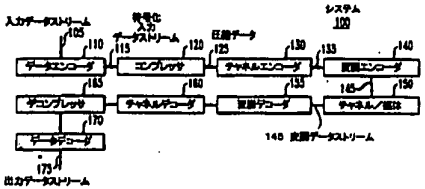
【符号の説明】

１０５　入力データストリーム

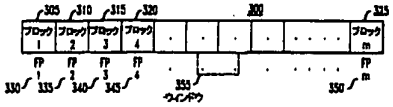
１１０　入力データエンコーダ

１１５　符号化入力データストリーム
１２０　コンプレッサ
１２５　圧縮データ
１３０　チャネルエンコーダ
１３５　チャネル符号化された情報
１４０　変調エンコーダ
１４５　変調データストリーム
１５０　媒体
１５５　変調デコーダ
１６０　チャネルデコーダ
１６５　デコンプレッサ
１７０　データデコーダ
１７５　出力データストリーム
２１０　入力データストリームを受信
２２０　圧縮ブロックサイズ「ｂ」を選択
２３０　入力データストリームをサイズ「ｂ」ブロック群に分割
２４０　各ブロックに対しフィンガプリントを計算し格納する
２５０　入力データストリームを横断し、キャラクタベースでフィンガプリントを計算し格納されたフィンガプリントと比較
２６０　マッチするか？
２７０　マッチを符号化
２８０　入力データストリームの終わりか？
２９０　符号化された入力データストリームに圧縮を適用
３９１、２４３９　文字位置
４１０　符号化出力データストリーム
４００、４２０～４５０　入力データストリーム
４６０～４９０　符号化出力
５００　入力データファイル
５１０、５２０、５３０　テキスト部分
６００　符号化データファイル
６１０、６２０、６３０　符号化部分
６３５　エンコーディング
７１０　符号化データファイルから文字「ｃ」を取り出す
７３０　出力ファイルに文字「ｃ」を書き込む
７４０　次の文字に進む。
「ｃ＝次の文字」
７６０　出力ファイルに「<」を書き込む
７７０　符号化データファイルから<start, length>エンコーディングを読み取る
７８０　start位置からlengthの長さのストリングを出力ファイルにコピーする
７９０　まだ「ｃ」はあるか？
９００　ソースプログラム
９１０、９２０、９３０、９４０、９５０　プログラムソースコード部分
１１００　ソースコードプログラム部分
１１２０　部分

【図１】



【図３】



【図４】

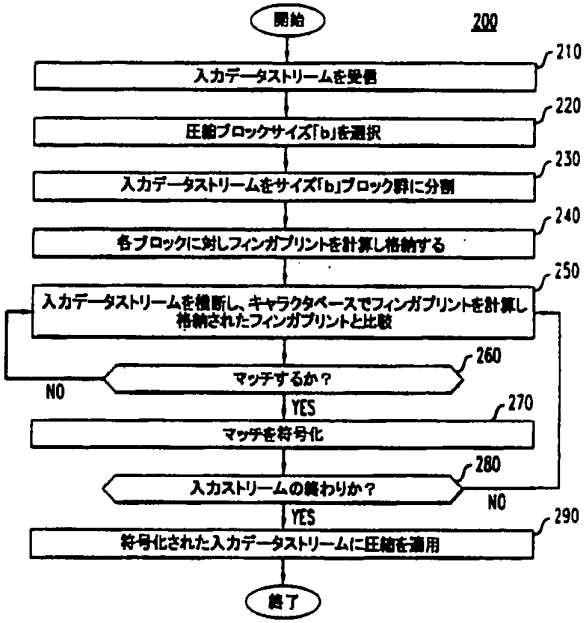
入力データストリーム　400	符号化出力データストリーム　410
abcdefghijklmnopq<12345>420	abcdefghijklmnopq<(12345>460
abcdefghijklmnopqrstuvwxyz 430	abcdefghijklmnopqrstuvwxyz<0,26><0,26><18,4><0,26><0,26>470
440	480
450	

【図８】

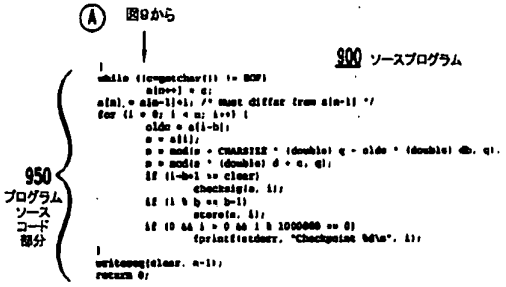
テキストファイルを圧縮した結果の比較 800

b (kbyte)	810 com b (kbyte)	820 com % (%)	830 com b / gzip (kbyte)	840	850 gzip % (%)	860 total % (%)
∞	68.123	100.00	22.925		34.64	34.64
4000	57.438	84.87	19.981		34.79	30.22
2000	56.886	83.15	19.842		34.82	30.01
1000	56.652	83.68	19.721		34.81	29.82
500	56.319	83.17	19.622		34.81	29.64
250	55.701	84.24	19.412		34.85	29.38
125	54.804	82.88	19.191		35.02	29.02
63	52.448	79.52	18.841		35.92	28.49
31	47.154	71.31	17.766		37.68	26.87
15	43.433	65.69	16.878		43.46	28.53
7	56.498	85.45	24.774		43.76	37.59

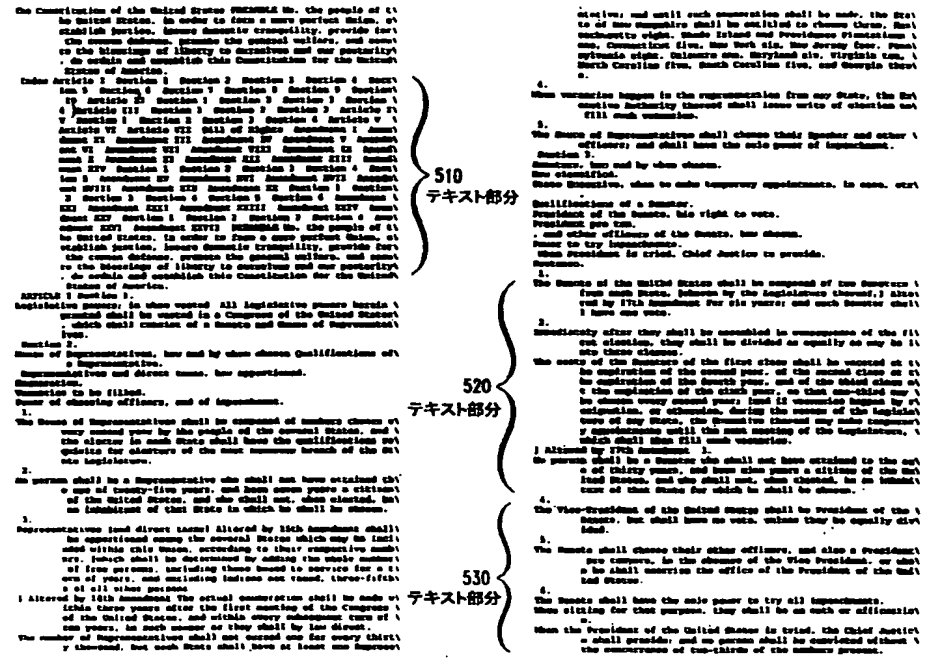
【図２】



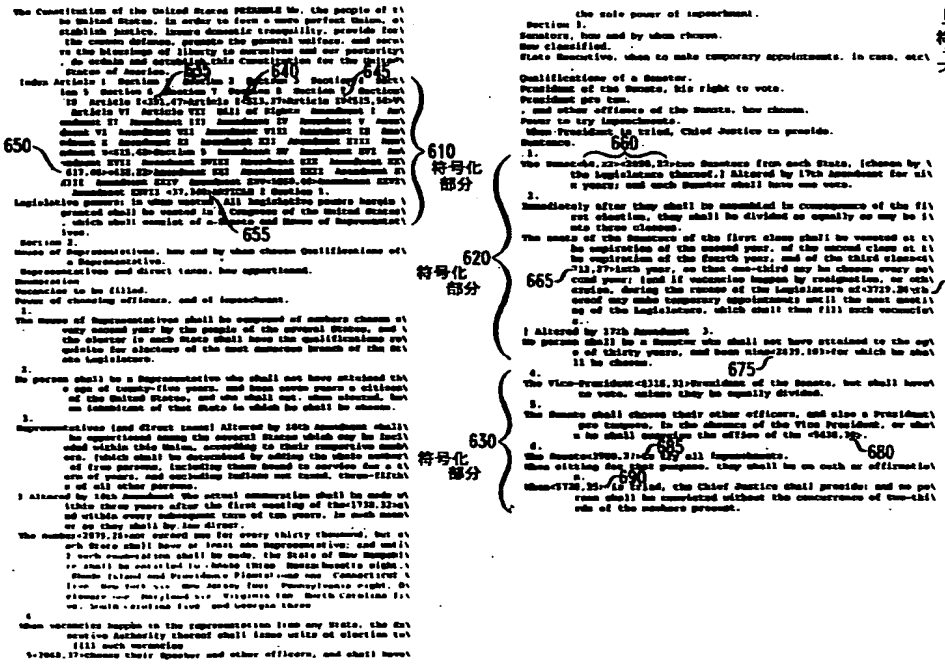
【図１０】



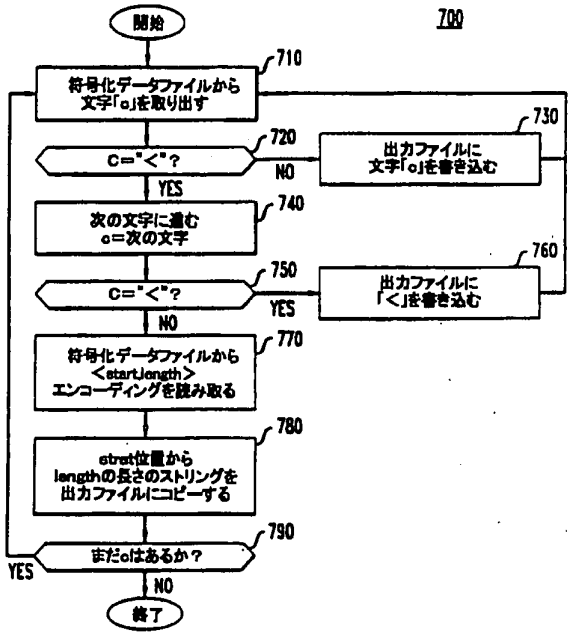
【図５】


500
入力データファイル
510
テキスト部分
520
テキスト部分
530
テキスト部分

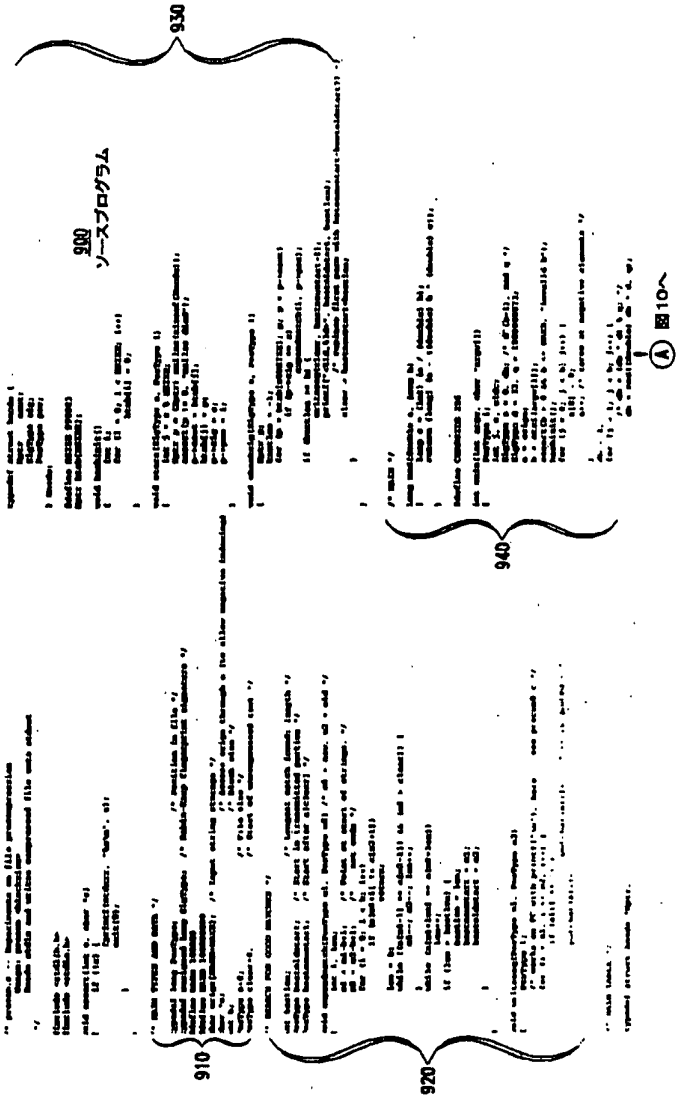
【図６】


600
符号化データファイル
610
符号化部分
620
符号化部分
630
符号化部分
650
655
660
665
670
675
680
685

【図7】


700
開始
710
符号化データファイルから
文字「o」を取り出す
720
C=="<"?
NO
730
出力ファイルに
文字「o」を書き込む
YES
740
次の文字に進む
o=次の文字
750
C=="<"?
YES
760
出力ファイルに
「<」を書き込む
NO
770
符号化データファイルから
<start,length>
エンコーディングを読み取る
780
strat位置から
lengthの長さのストリングを
出力ファイルにコピーする
790
まだoはあるか?
YES
NO
終了

【図9】


900
ソースプログラム
910
920
930
940
A
図10へ

【図１１】

1100 ソースコードプログラム部分

```
/* decom.c -- decompress files made by precom.c
    Usage: precom
    Reads stdin and writes decompressed file onto stdout
 */

#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>

typedef long PosType;
#define MAXN 100000000
char a[MAXN];    /* Output string storage */
PosType n=0;     /* File size */

void addchar(char c)
{   a[n++] = c;
    putchar(c);
}

int getint() /* gobble nondigit at end */
{   int c, n = 0;
    for (;;) {
        c = getchar(); /* danger: no EOF */
        if (!isdigit(c)) break;
        n = 10*n + c - '0';
    }
    return n;
}

int main()
{   PosType i;
    int c, start, len;
    while ((c=getchar()) != EOF) {
        if (c == '<') {
            c = getchar(); /* danger: no EOF */
            if (c == '<') { /* quoted < */
                addchar('<');
            } else {
                ungetc(c, stdin);
                start = getint();
                len = getint();
                for (i = start; i < start+len; i++)
                    addchar(a[i]);
            }
        } else
            addchar(c);
    }
    return 0;
```

1110

1120

フロントページの続き

(71)出願人 596077259
600 Mountain Avenue, Murray Hill, New Jersey 07974-0636U. S. A.

(72)発明者 ジョン ルイス ベントリー
アメリカ合衆国、07974 ニュージャージー、ニュープロビデンス、セントラル アベニュー 449, アパートメント 2

(72)発明者 マルコルム ダグラス マクローイ
アメリカ合衆国、03750 ニューハンプシャー、エトナ、ヘイフィールド ロード 1